FIGURE 1

90

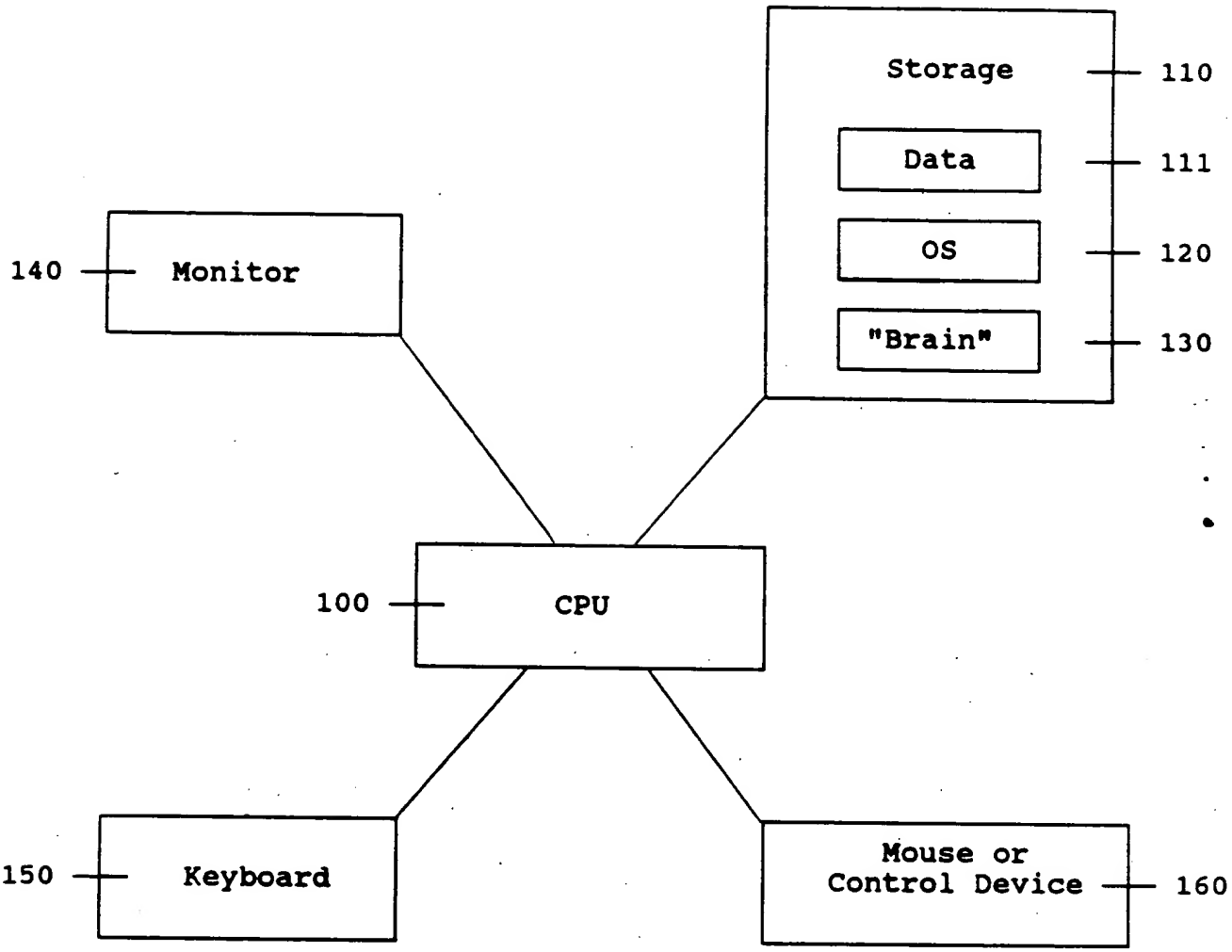


FIGURE 2

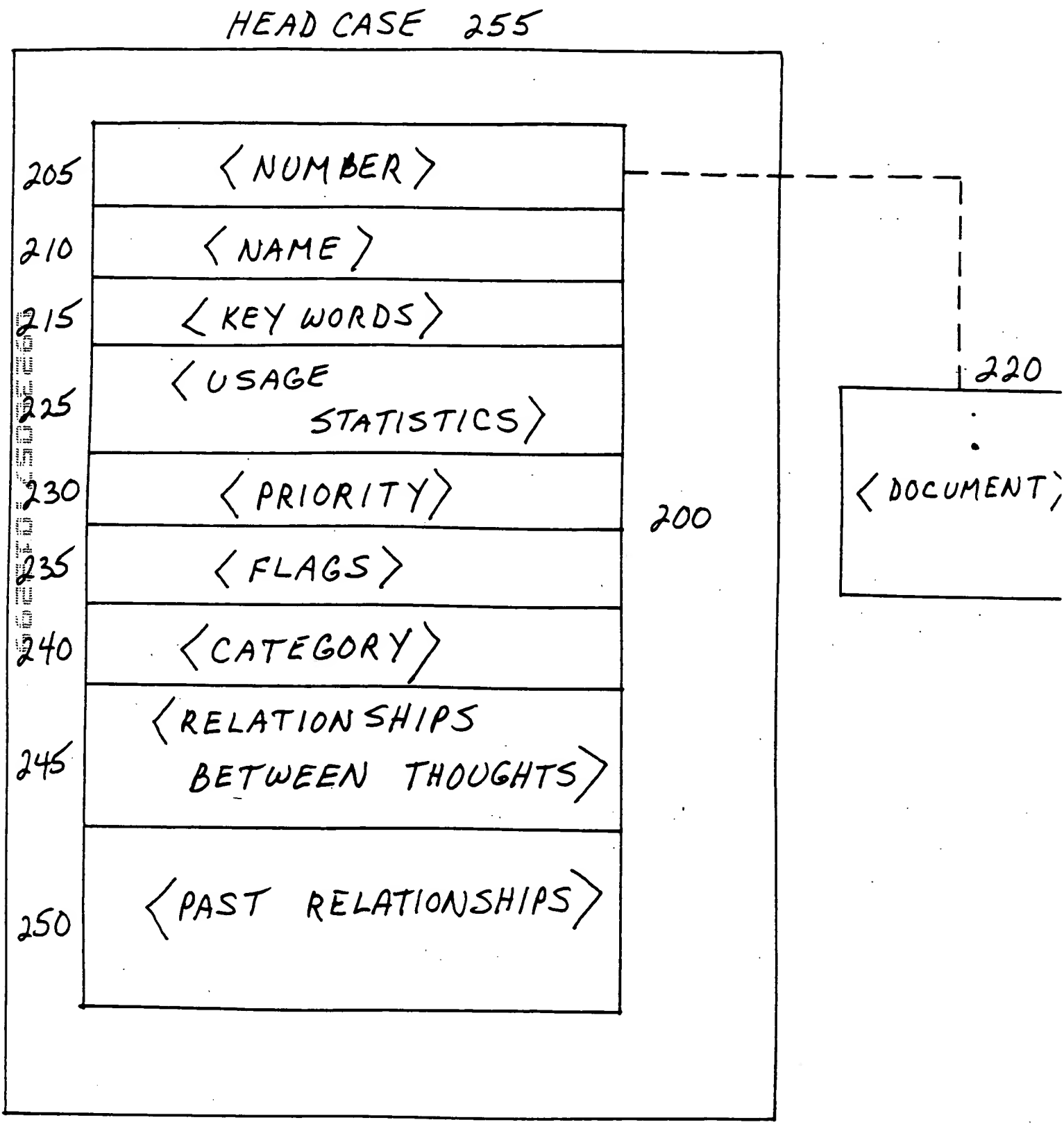

HEAD CASE 255
205
<NUMBER>
210
<NAME>
215
<KEY WORDS>
225
<USAGE STATISTICS>
230
<PRIORITY>
235
<FLAGS>
240
<CATEGORY>
245
<RELATIONSHIPS BETWEEN THOUGHTS>
250
<PAST RELATIONSHIPS>
200
220
<DOCUMENT>

FIGURE 3

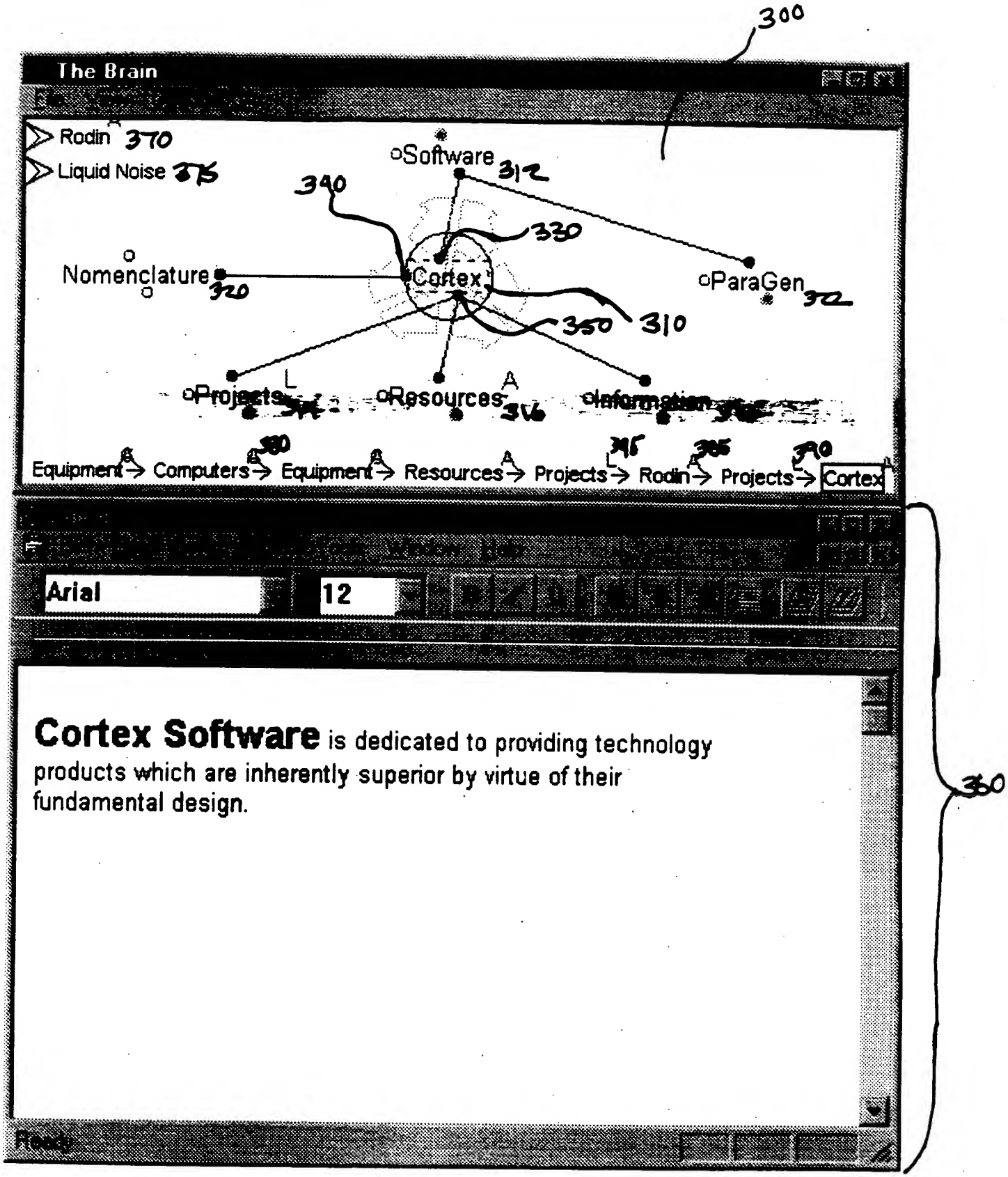

300
The Brain
Rodin 370
Liquid Noise 375
Software 312
340
330
Nomenclature 320
Cortex
ParaGen 322
350
310
Projects
Resources 316
Information
Equipment → Computers → Equipment → Resources → Projects → Rodin → Projects → Cortex
Arial
12
Cortex Software is dedicated to providing technology products which are inherently superior by virtue of their fundamental design.
Ready
360

FIGURE 4

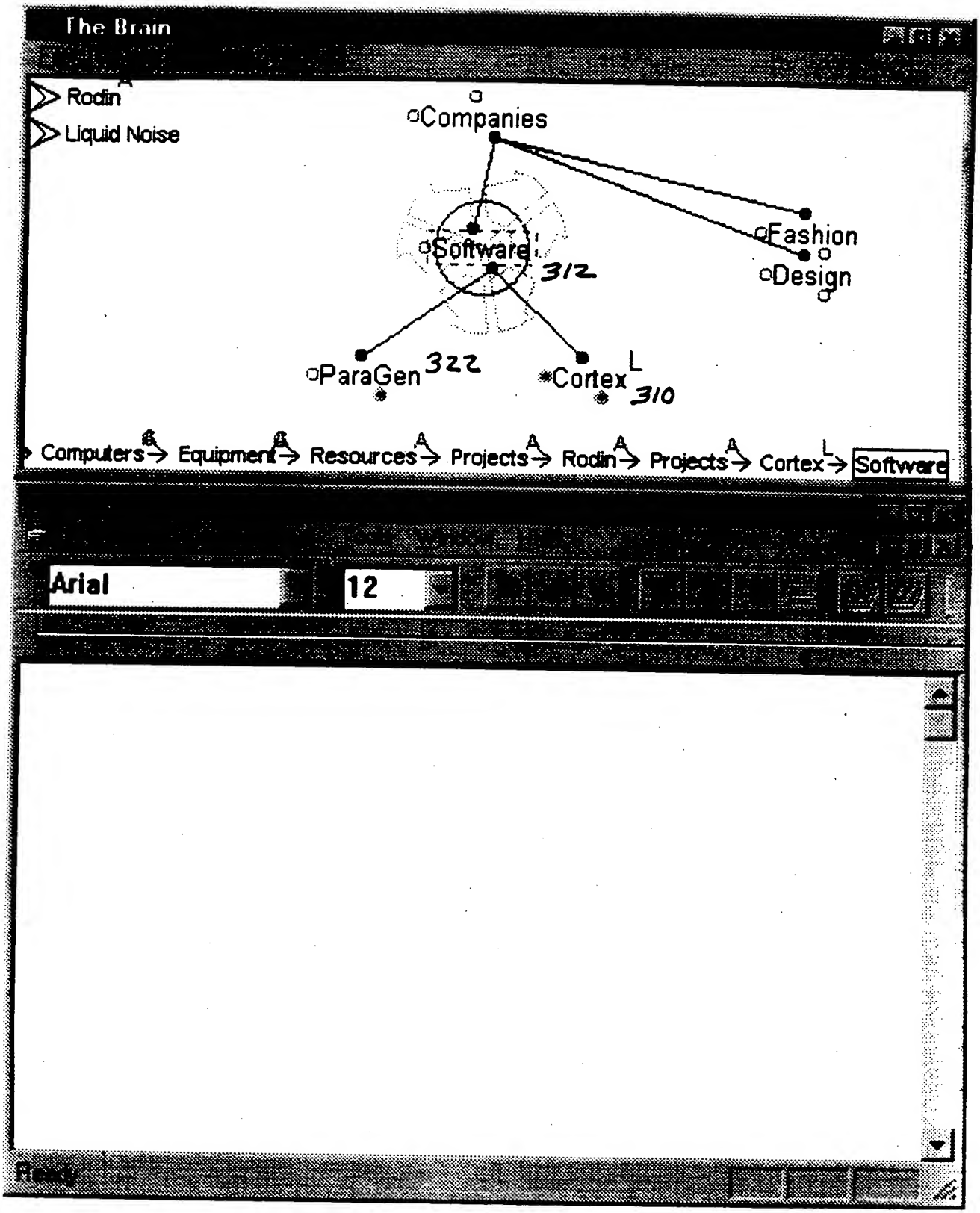

The Brain
Rodin
Liquid Noise
Companies
Software
312
Fashion
Design
ParaGen
322
Cortex
310
Computers
Equipment
Resources
Projects
Rodin
Projects
Cortex
Software
Arial
12
Ready

FIGURE 5

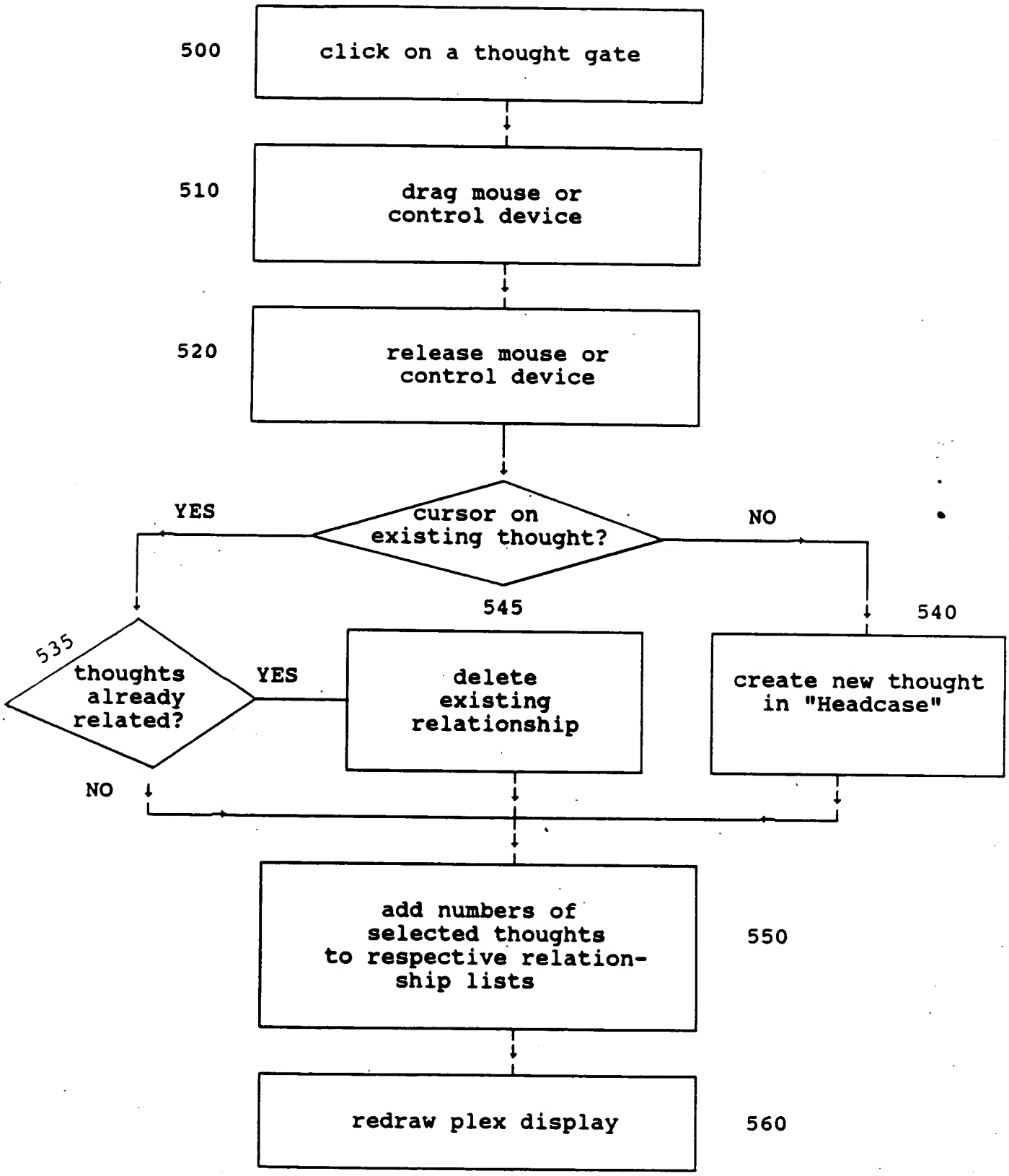


Fig. 6

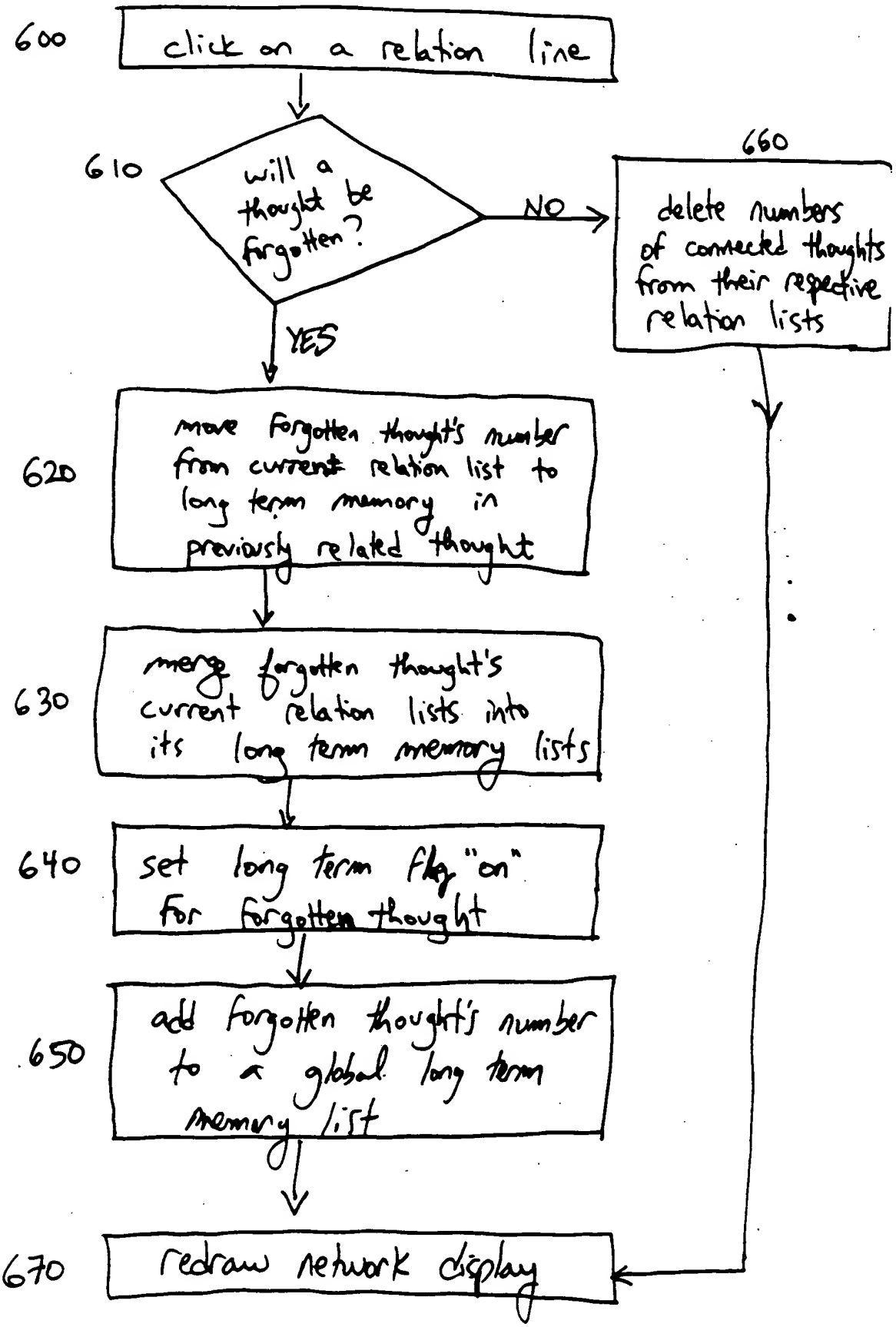

600
click on a relation line
610
will a thought be forgotten?
NO
660
delete numbers of connected thoughts from their respective relation lists
YES
620
move forgotten thought's number from current relation list to long term memory in previously related thought
630
merge forgotten thought's current relation lists into its long term memory lists
640
set long term flag "on" for forgotten thought
650
add forgotten thought's number to a global long term memory list
670
redraw network display

FIGURE 7

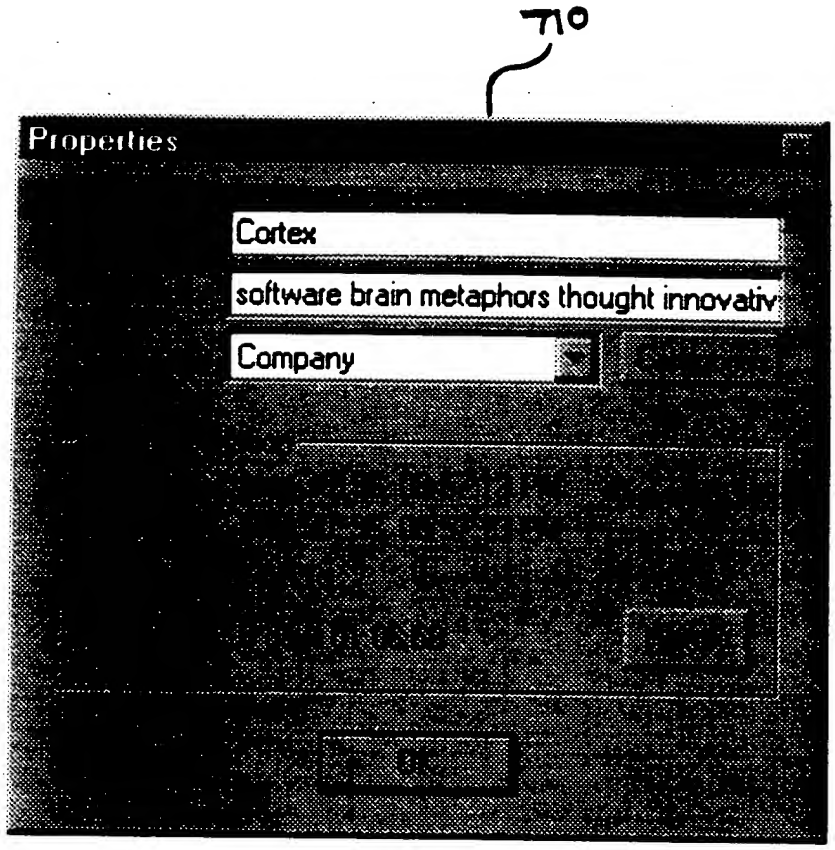

710
Properties
Cortex
software brain metaphors thought innovativ
Company
OK

FIGURE 8

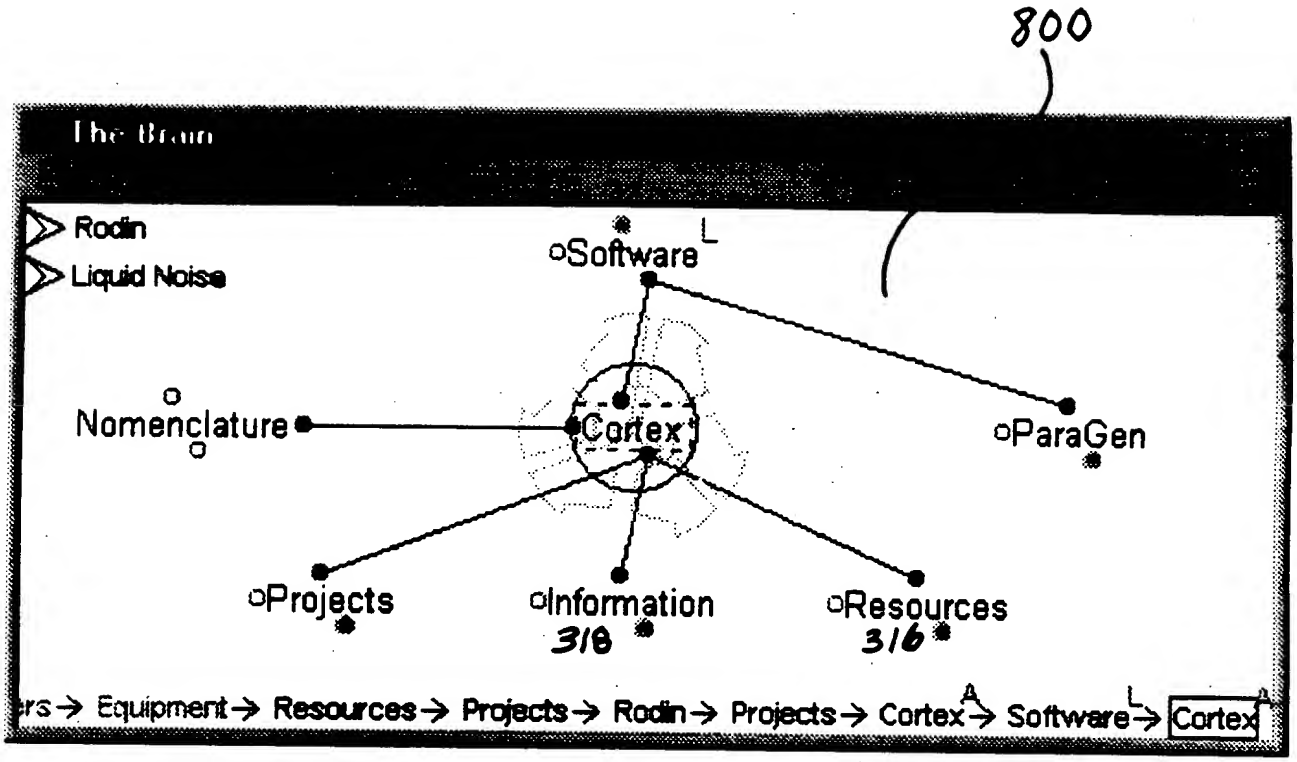

800
The Brain
Rodin
Liquid Noise
Software
Nomenclature
Cortex
ParaGen
Projects
Information
318
Resources
316
rs→ Equipment→ Resources→ Projects→ Rodin→ Projects→ Cortex→ Software→ Cortex

FIGURE 9

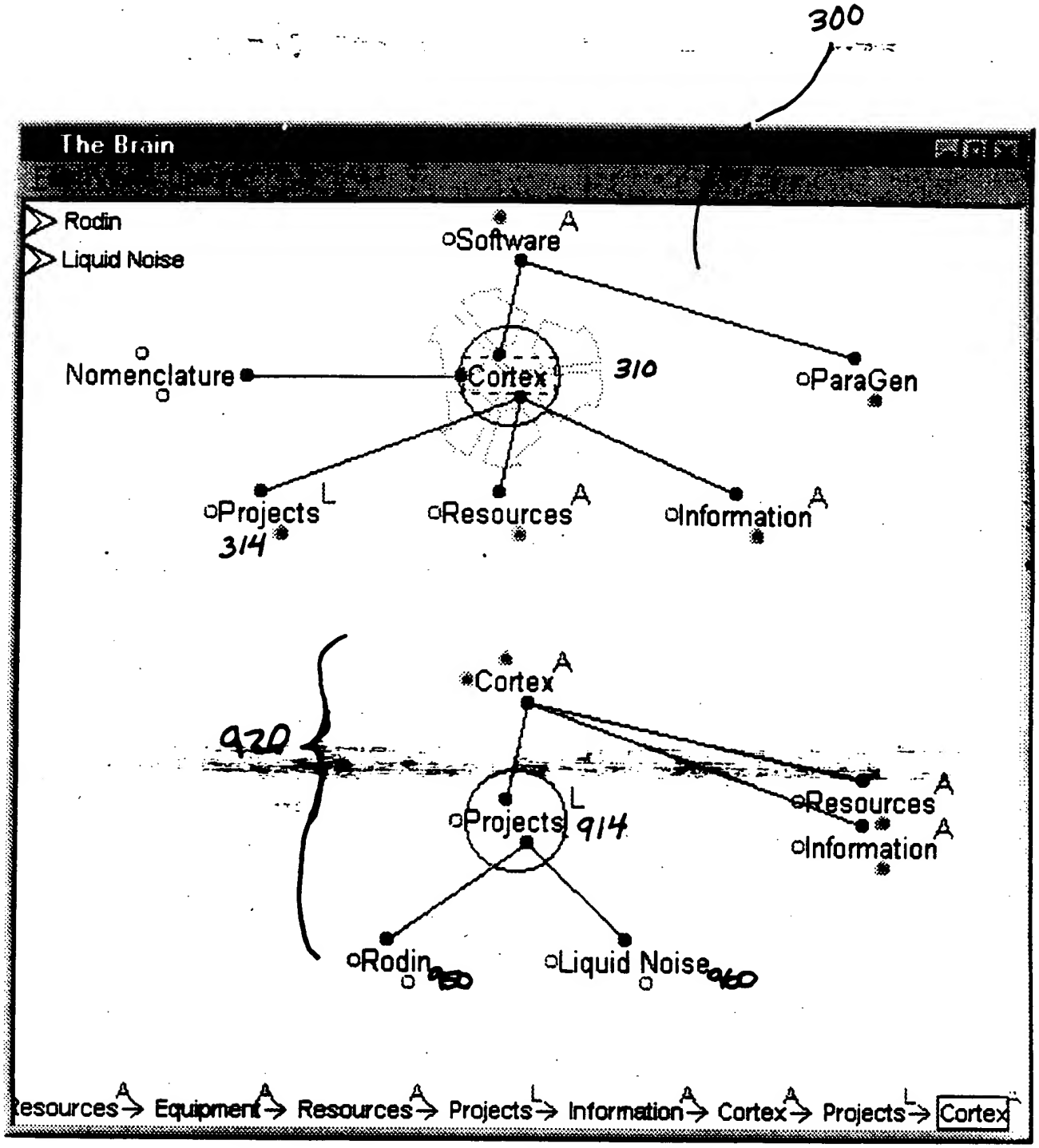

300
The Brain
Rodin
Liquid Noise
Software
A
Nomenclature
Cortex
310
ParaGen
Projects
L
314
Resources
A
Information
A
Cortex
A
920
Resources
A
Projects
L
914
Information
A
Rodin
950
Liquid Noise
960
Resources
A
Equipment
A
Resources
A
Projects
L
Information
A
Cortex
A
Projects
L
Cortex

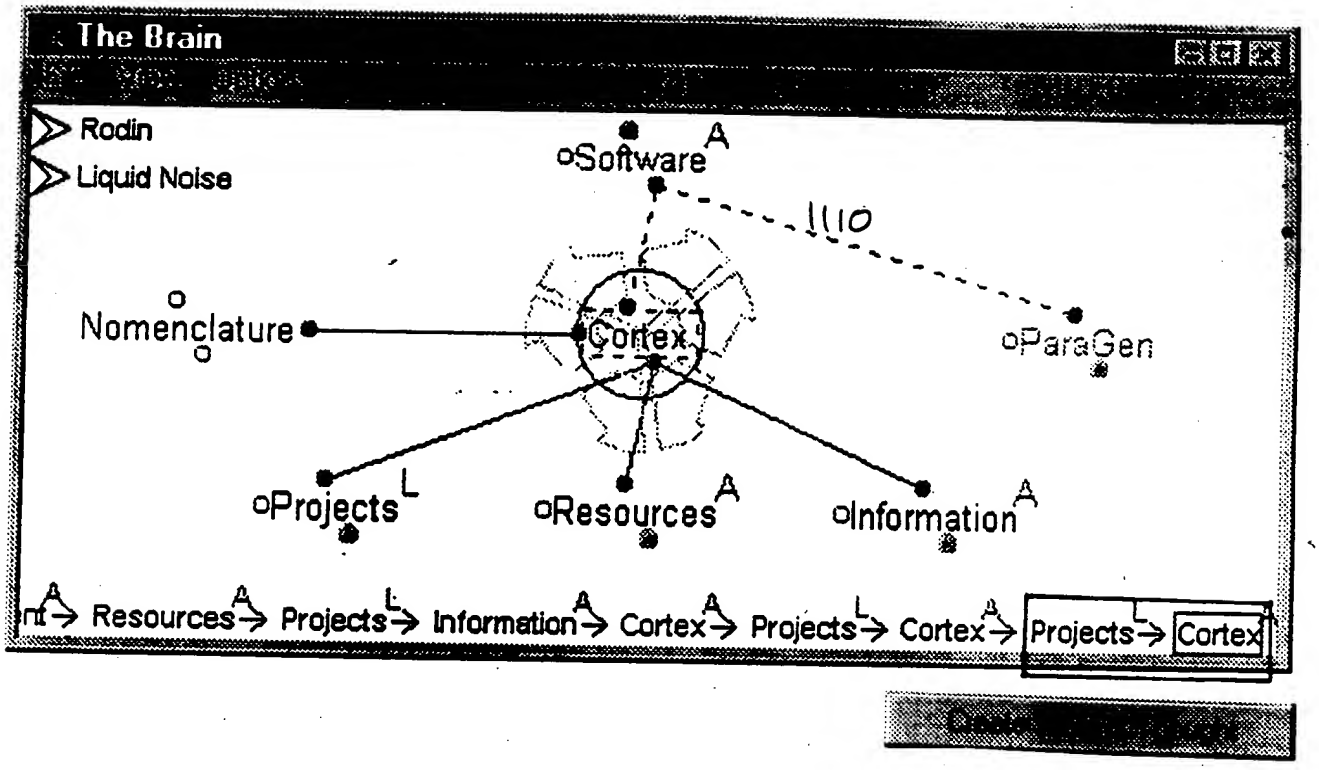
Figure 10

```
boolean CheckForIsolation(int centralThought, int targetThought)
{

        // this function checks if centralThought is related to targetThought
        // via any of targetThought's relations (not directly)

        // remove centralThought as a direct relation from targetThought
        RemoveRelation(targetThought, centralThought);

        // create an empty thought list to keep track of the search
        intList searchList = CreateEmptyList();

        // start recursive searches on each of targetThought's direct relations
        int relation = GetFirstRelation(targetThought);
        boolean found;
        do {
                found = Search(relation, centralThought, searchList);
                if(found) {
                        // centralThought was found, no need to search any further
                        break;
                }
                // this loop will end when there are no more relations
        } while(relation = GetNextRelation(targetThought);

        // add centralThought back onto targetThought as a relation
        AddRelation(targetThought, centralThought);

        return found;

}
```

Figure 10, cont'd

```
boolean Search(source, dest, searchList)
{

        if(Find(source, searchList)) {
                // source has already been searched
                return FALSE;
        }

        // add source to the searchList
        Add(source, searchList)

        if(source == dest) {
                // this is the destination, we have found it
                return TRUE;
        }

        // recursive searches on each of sources direct relations
        int relation = GetFirstRelation(source);
        boolean found;
        do {
                found = Search(relation, dest, searchList);
                if(found) {
                        // centralThought was found, no need to search any further
                        break;
                }
                // this loop will end when there are no more relations
        } while(relation = GetNextRelation(targetThought);

        return found;

}
```

FIGURE 11



1120

FIGURE 12

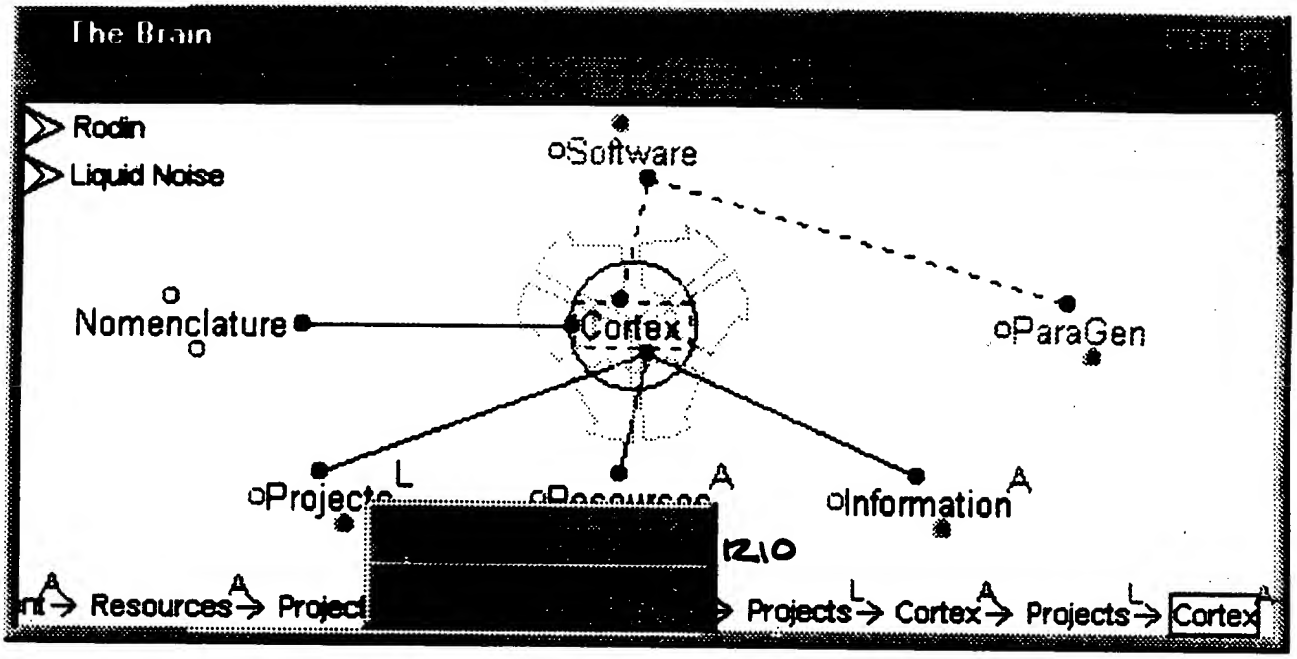

The Brain
Rodin
Liquid Noise
Software
Nomenclature
Cortex
ParaGen
Project
Information
Resources
Projects
Cortex
Projects
Cortex

FIGURE 13

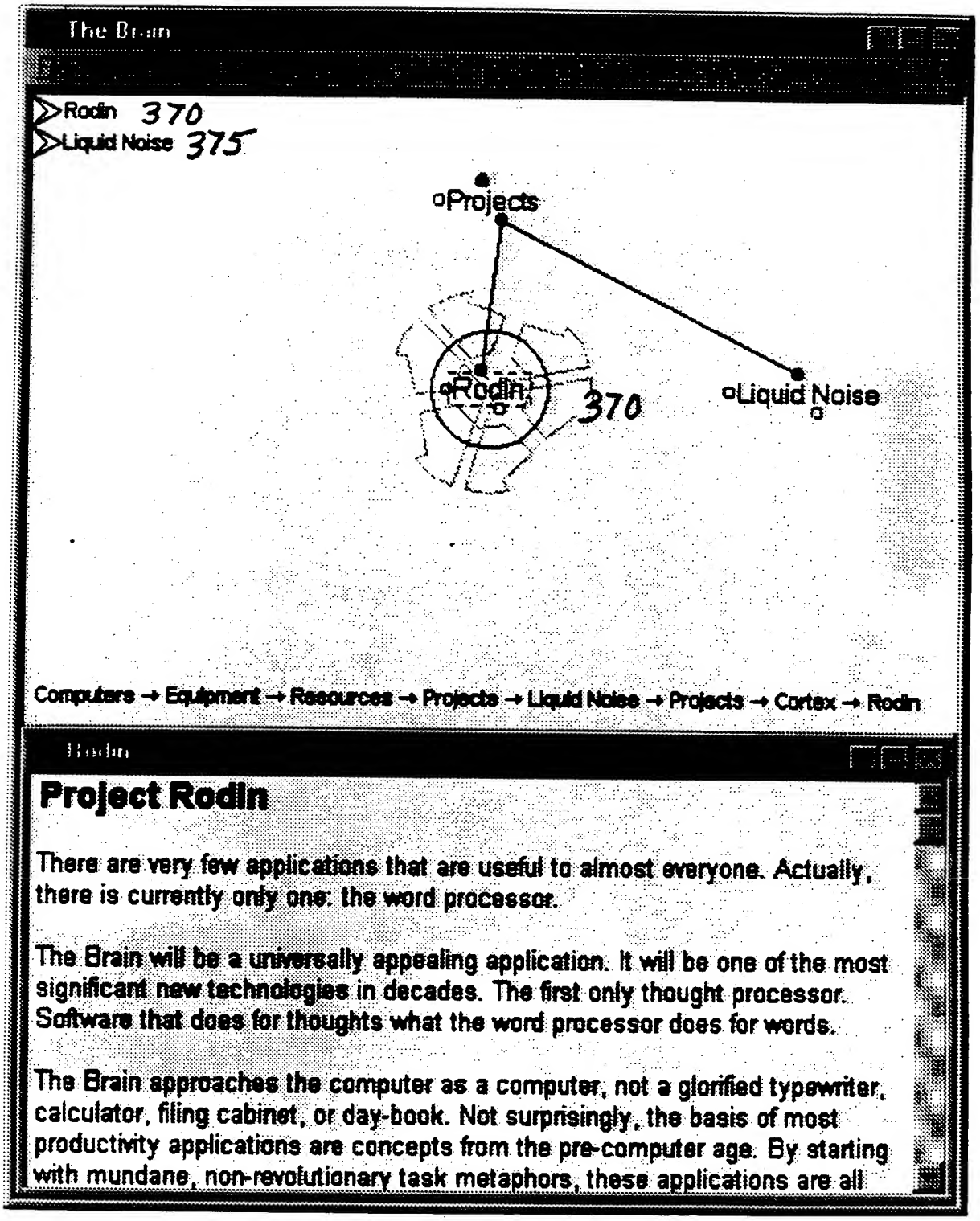

The Brain
Rodin 370
Liquid Noise 375
Projects
Rodin
370
Liquid Noise
Computers → Equipment → Resources → Projects → Liquid Noise → Projects → Cortex → Rodin
Rodin
Project Rodin
There are very few applications that are useful to almost everyone. Actually, there is currently only one: the word processor.
The Brain will be a universally appealing application. It will be one of the most significant new technologies in decades. The first only thought processor. Software that does for thoughts what the word processor does for words.
The Brain approaches the computer as a computer, not a glorified typewriter, calculator, filing cabinet, or day-book. Not surprisingly, the basis of most productivity applications are concepts from the pre-computer age. By starting with mundane, non-revolutionary task metaphors, these applications are all

FIGURE 14

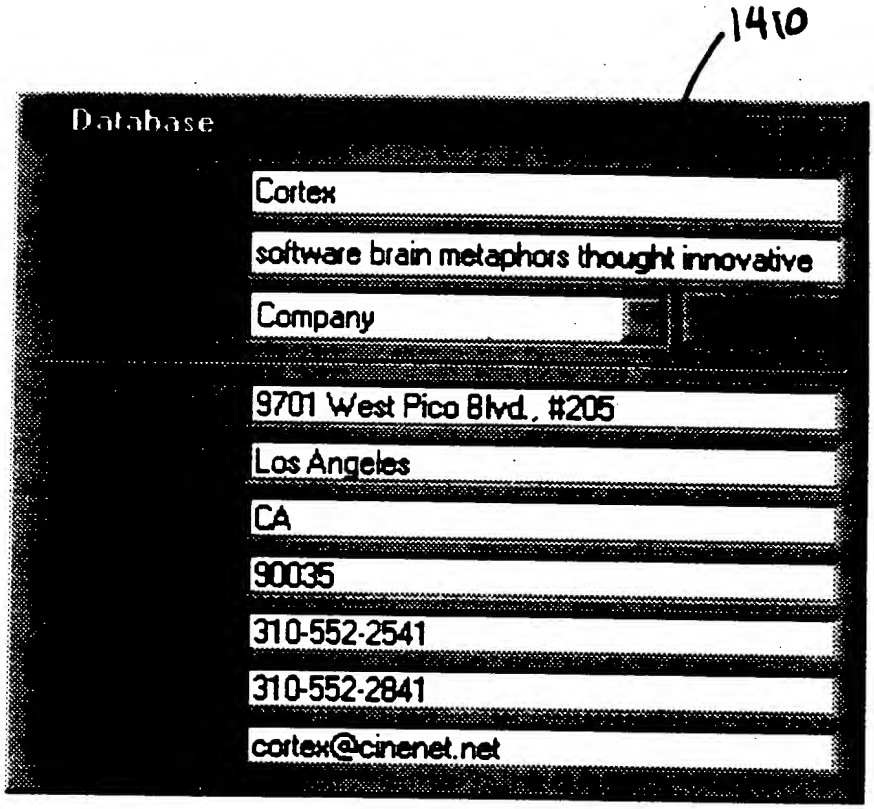
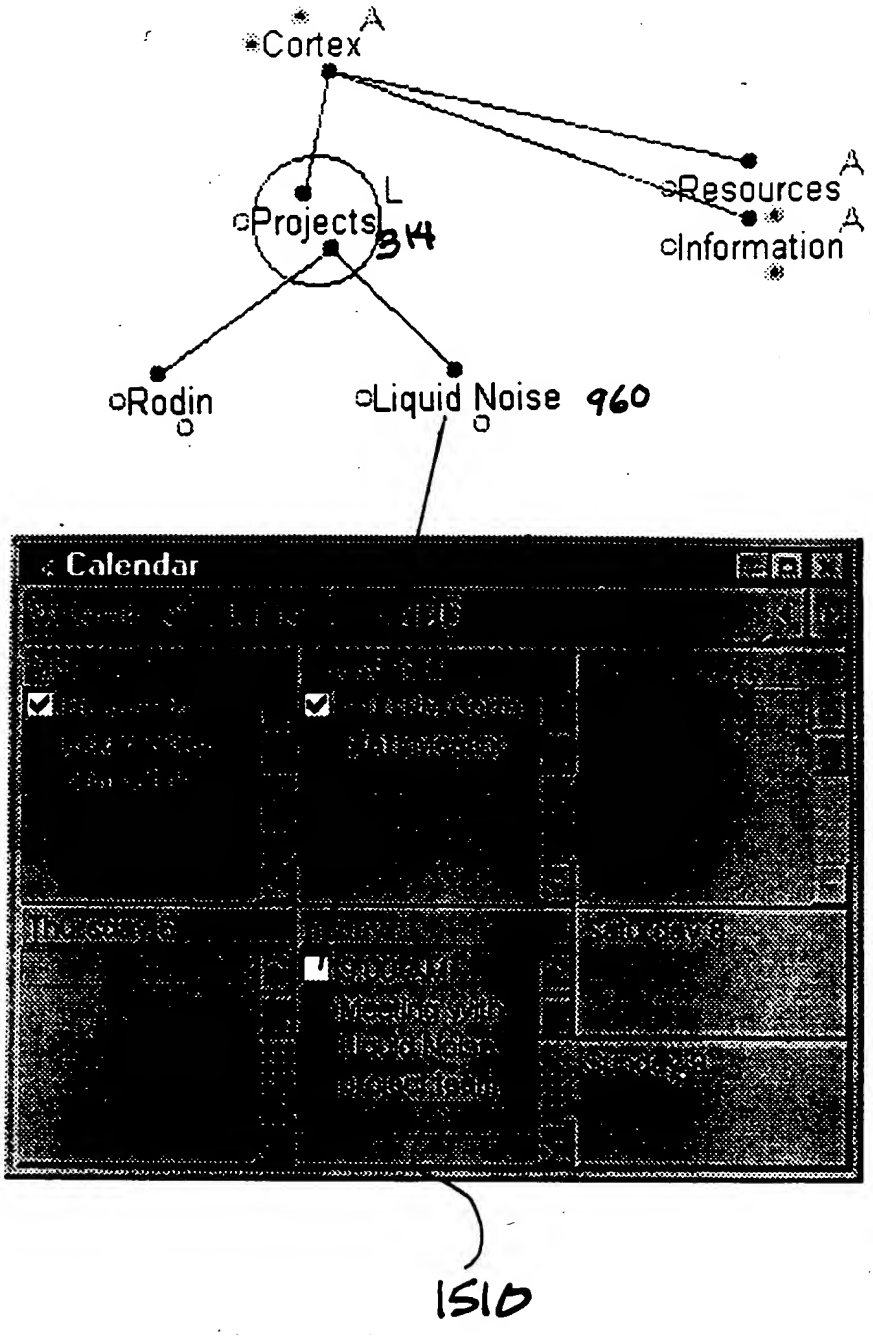

1410
Database
Cortex
software brain metaphors thought innovative
Company
9701 West Pico Blvd., #205
Los Angeles
CA
90035
310-552-2541
310-552-2841
cortex@cinenet.net

FIGURE 15


Cortex
A
Resources
A
Information
A
Projects
L
34
Rodin
Liquid Noise
960
Calendar
1510

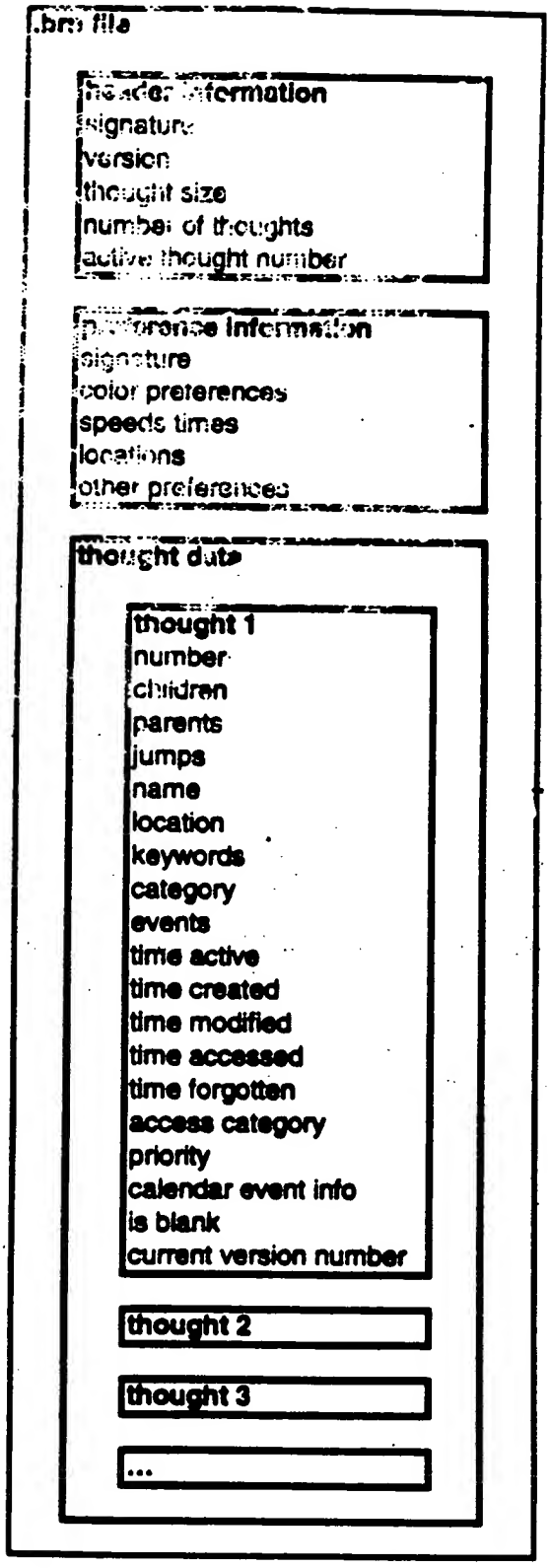

.brn file
header information
signature
version
thought size
number of thoughts
active thought number
preference information
signature
color preferences
speeds times
locations
other preferences
thought data
thought 1
number
children
parents
jumps
name
location
keywords
category
events
time active
time created
time modified
time accessed
time forgotten
access category
priority
calendar event info
is blank
current version number
thought 2
thought 3
...

Fig. 16

```
ForgetThought(fNum)
{
  // mark all the children of the selected thought
  list.Clear();
  MarkChildren(fNum, list);
  // unmark the active thought
  list.RemoveThought(activeThought);
  // unmark thoughts with unmarked parents
  lNum = list.GetFirstNum();
  while(lNum != 0)
  {
    if(lNum != fNum)  // don't unmark the selected thought
    {
      pNum = GetFirstThoughtParent(lNum);
      while(pNum != 0)
      {
        if(list.Contains(pNum) == FALSE)
        {
          if(IsThoughtInLongTermMemory(pNum) == FALSE)
          {
            // unmark all the children of the unmarked parent
            childList.Clear();


            MarkChildren(pNum, childList);
            list.RemoveList(childList);
          }
        }
        pNum = GetNextThoughtParent(lNum);
      }
    }
    lNum = list.GetNextNum();
  }
  // now forget all the thoughts left on the list
  lNum = list.GetFirstNum();
  while(lNum != 0)
  {
    ForgetThought(lNum);
    lNum = list.GetNextNum();
  }
}

RememberThought(rNum)
{
  // mark all the children of the selected thought
  list.Clear();
  MarkChildren(rNum, list);
  // remember all the thoughts on the list
  lNum = list.GetFirstNum();
  while(lNum != 0)
  {
    RemeberThought(lNum);
    lNum = list.GetNextNum();
  }
}

MarkChildren(num, list)
{
  list.AddThought(num);
  cNum = GetFirstChild(num);
  while(cNum != 0)
  {
    MarkChildren(cNum, list);
    cNum = GetNextChild(num);
  }
}
```

Fig. 17

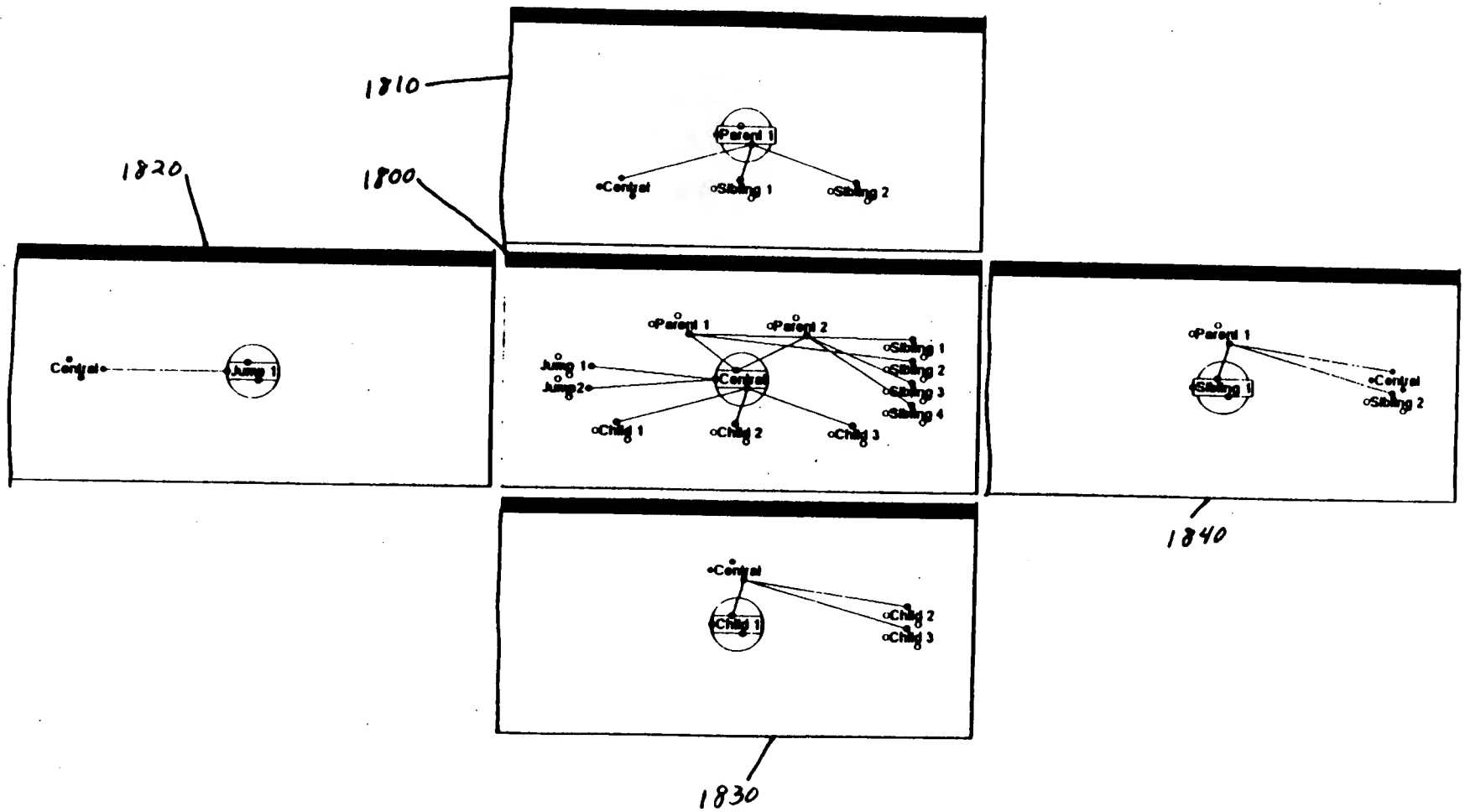

1810
1820
1800
Parent 1
Central
Sibling 1
Sibling 2
Central
Jump 1
Parent 1
Parent 2
Sibling 1
Sibling 2
Sibling 3
Sibling 4
Jump 1
Jump 2
Central
Child 1
Child 2
Child 3
Parent 1
Sibling 1
Central
Sibling 2
1840
Central
Child 1
Child 2
Child 3
1830

Figure 18

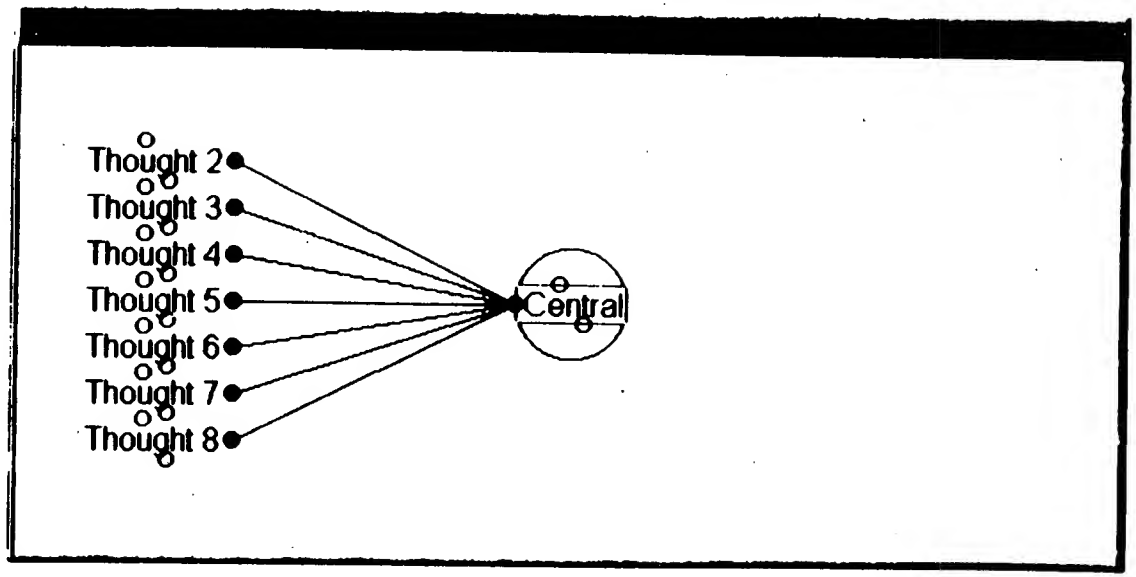

Thought 2
Thought 3
Thought 4
Thought 5
Thought 6
Thought 7
Thought 8
Central

Figure 19

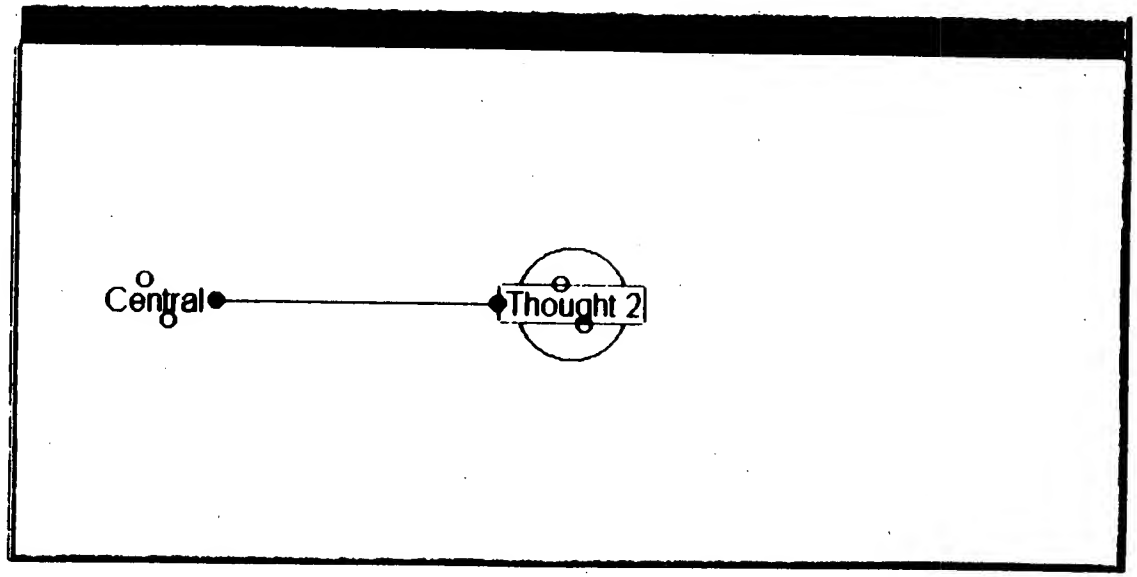


Figure 20

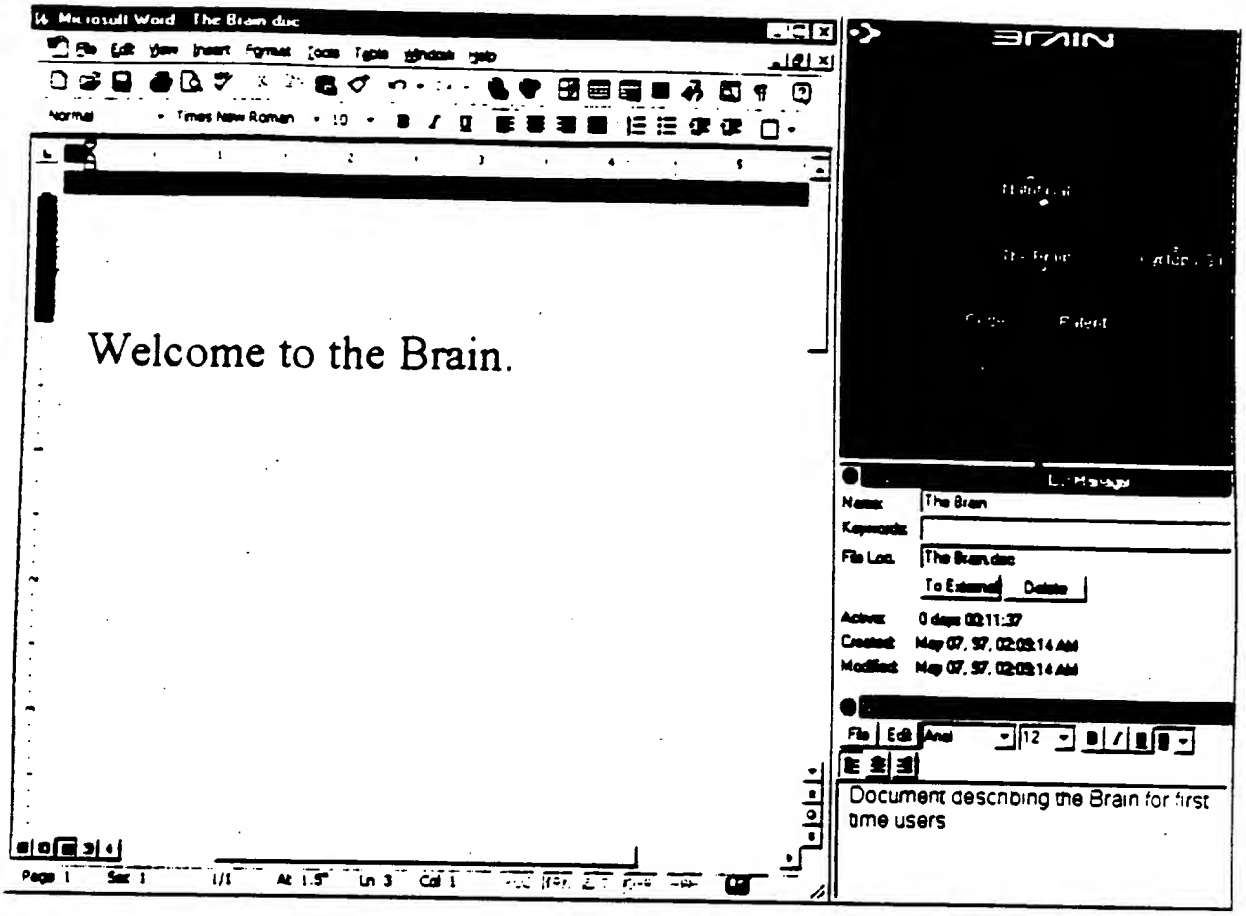

Welcome to the Brain.
BRAIN
The Brain
The Brain.doc
Document describing the Brain for first time users

Figure 21

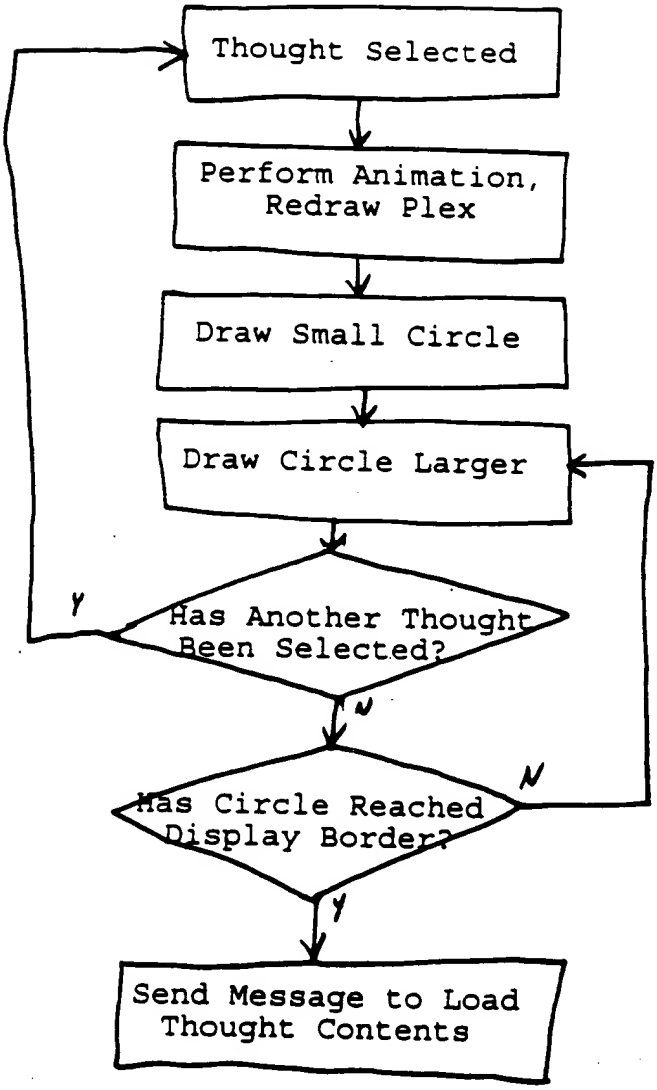


Fig. 22

Algorithm for drawing the plex with distant thoughts

```
1.      Create a list of thoughts to be drawn and their on screen locations:
2.          Add the central thought to the list.
3.          Add children to the list.
4.          Add parents to the list.
5.          Add jumps to the list.
6.          Add siblings to the list, checking first that they are not
                already on the list.
7.          Add distants of children to the list, checking first that they
                are not already on the list.
8.          Add distants of parents to the list, checking first that they
                are not already on the list.
9.          Add distants of jumps to the list, checking first that they
                are not already on the list.
10.         Add distants of siblings to the list, checking first that
                they are not already on the list.
11.     Draw the lines that connect each thought:
12.         For each item in the list:
13.             Get each item in the list:
14.                 If the two items are related, draw lines
                        between them from and to the appropriate
                        gates.
15.     Draw the distant thoughts:
16.         For each item in the list:
17.             If it is a distant thought, draw it.
18.     Draw the other thoughts:
19.         For each item in the list:
20.             If it is not a distant thought, draw it.
```

Figure 23

```
// the non [redacted]ve method for searching thoughts
// tries to fi[redacted]route from nSrc to nDest other than a direct relation
// returns TRUE if found
boolean Search(int nSrc, int nDest)
{
        // create the lists
        ThoughtList posList;    // list of thoughts that possibly connect
        ThoughtList notList;    // list of thought that do not connect
        // empty the lists
        posList.Initialize();
        notList.Initialize();

        // add the source to the not list since we cannot go directly to the destination
        notList.Add(nSrc);

        // since we cannot go directly to the destination,
        // add all relates except the destination to the possible list
        Thought src(nSrc);
        for(int n = 0; ; n++)
        {
                int nRel = src.GetRelate(n);
                if(!nRel)
                {
                        // no more relations, done
                        break;
                }
                if(nRel != nDest)
                {
                        // add it to the possibly connect list
                        posList.Add(nRel);
                }
        }

        while(TRUE)
        {
                // check the first possibility
                int nTest = posList.GetFirst();
                if(!nTest)
                {
                        // nothing on the list, done
                        break;
                }
                Thought test(nTest);
                if(test.IsRelated(nDest))
                {
                        // this one is related to the destination, we're done
                        return TRUE;
                }
                // does not connect, add it to the does not connect list
                notList.Add(nTest);
                // add all related thoughts except those already checked to possible list
                for(int n = 0; ; n++)
                {
                        int nRel = test.GetRelate(n);
                        if(!nRel)
                        {

                                // no more relations, done
                                break;
                        }
                        if(!notList.Exists(nRel))
                        {
                                // not checked yet, add to possible list
                                posList.Add(rel);
                        }
                }
                // remove this one from the possible list
                posList.Remove(nTest);
        }
        // we've checked everything there is no other way to get from nSrc to nDest
        return FALSE;
}
```

Figure 24

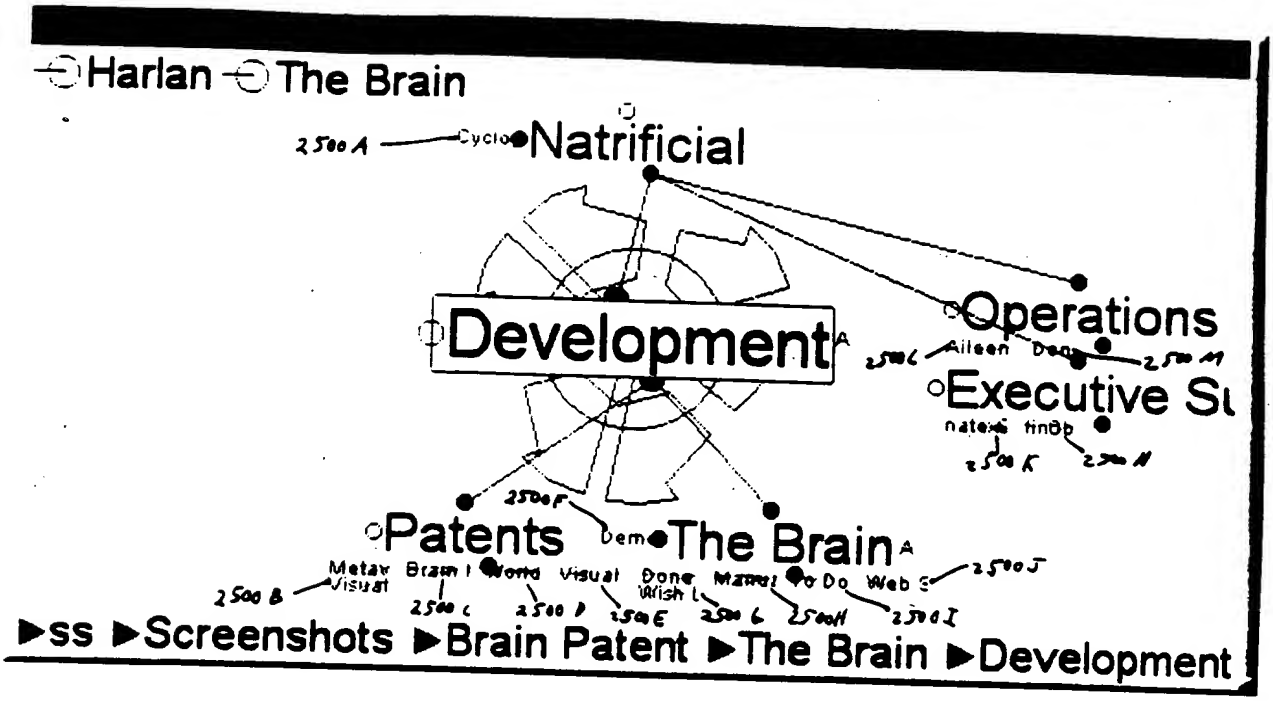

Harlan
The Brain
Natrificial
Development
Operations
Executive
Patents
The Brain
Screenshots
Brain Patent
The Brain
Development

Figure 25

Figure 26

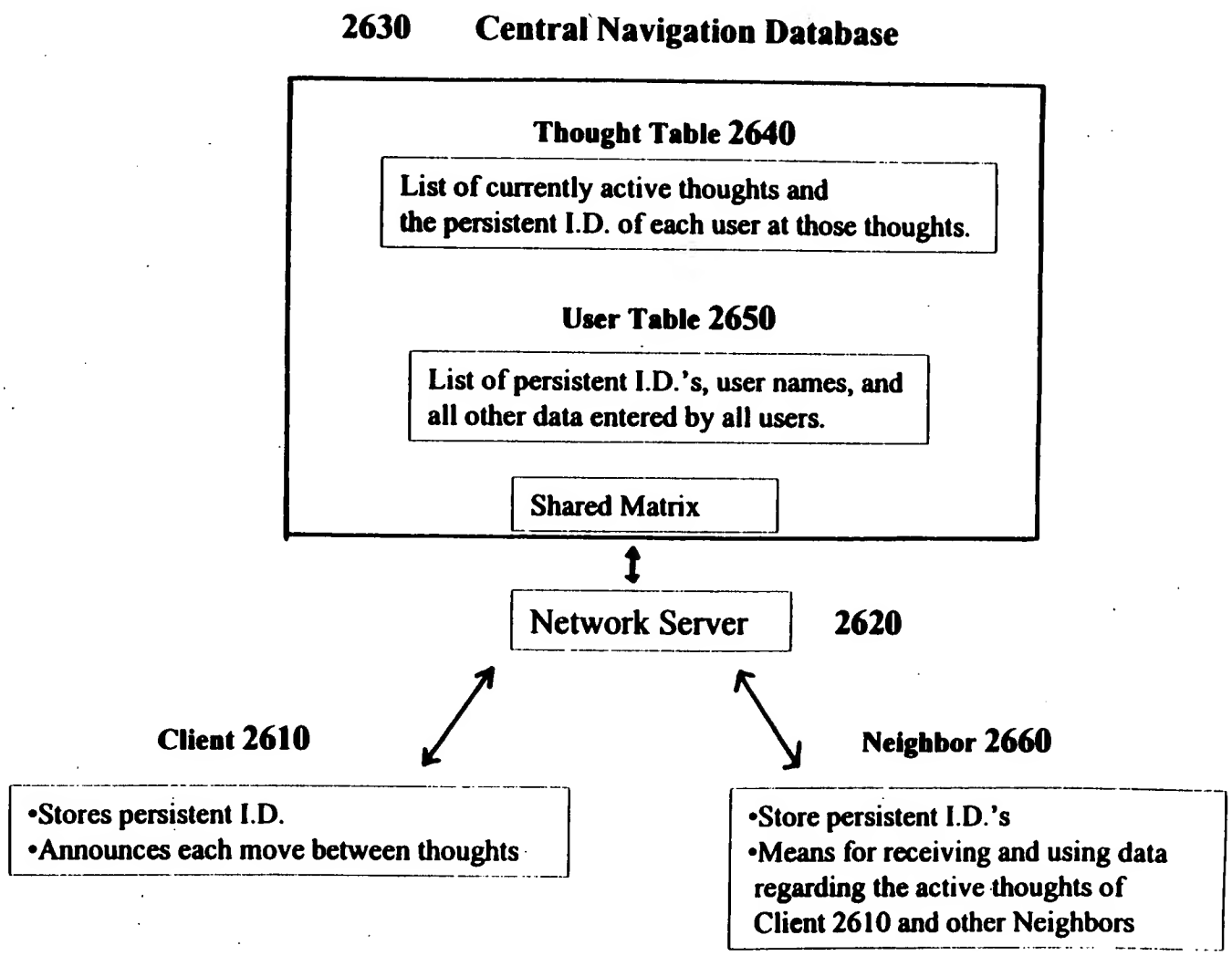

2630 Central Navigation Database
Thought Table 2640
List of currently active thoughts and the persistent I.D. of each user at those thoughts.
User Table 2650
List of persistent I.D.'s, user names, and all other data entered by all users.
Shared Matrix
Network Server
2620
Client 2610
•Stores persistent I.D.
•Announces each move between thoughts
Neighbor 2660
•Store persistent I.D.'s
•Means for receiving and using data regarding the active thoughts of Client 2610 and other Neighbors

Figure 27

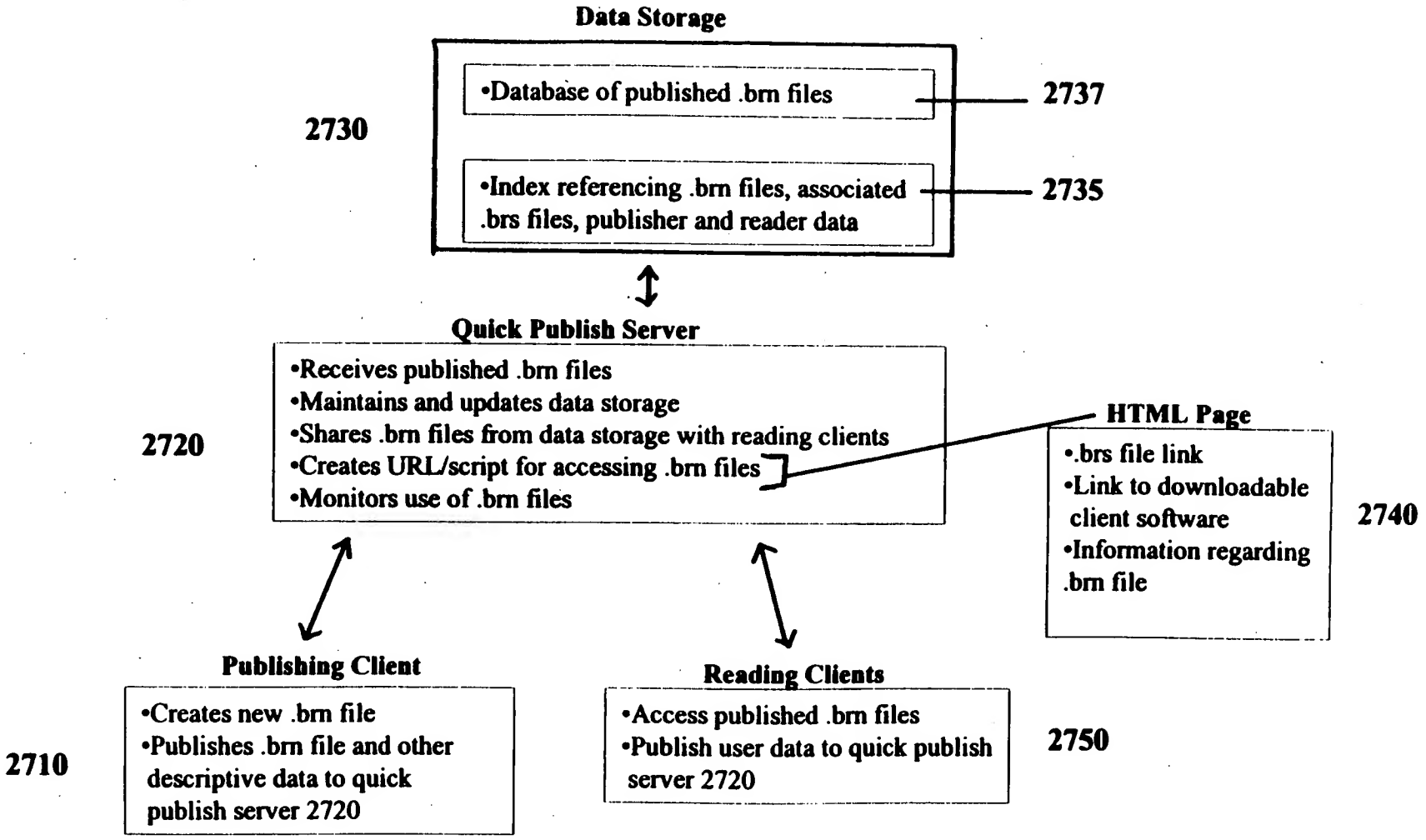

Data Storage
•Database of published .brn files
2737
2730
•Index referencing .brn files, associated .brs files, publisher and reader data
2735
Quick Publish Server
•Receives published .brn files
•Maintains and updates data storage
•Shares .brn files from data storage with reading clients
•Creates URL/script for accessing .brn files
•Monitors use of .brn files
2720
HTML Page
•.brs file link
•Link to downloadable client software
•Information regarding .brn file
2740
Publishing Client
•Creates new .brn file
•Publishes .brn file and other descriptive data to quick publish server 2720
2710
Reading Clients
•Access published .brn files
•Publish user data to quick publish server 2720
2750

FIGURE 28

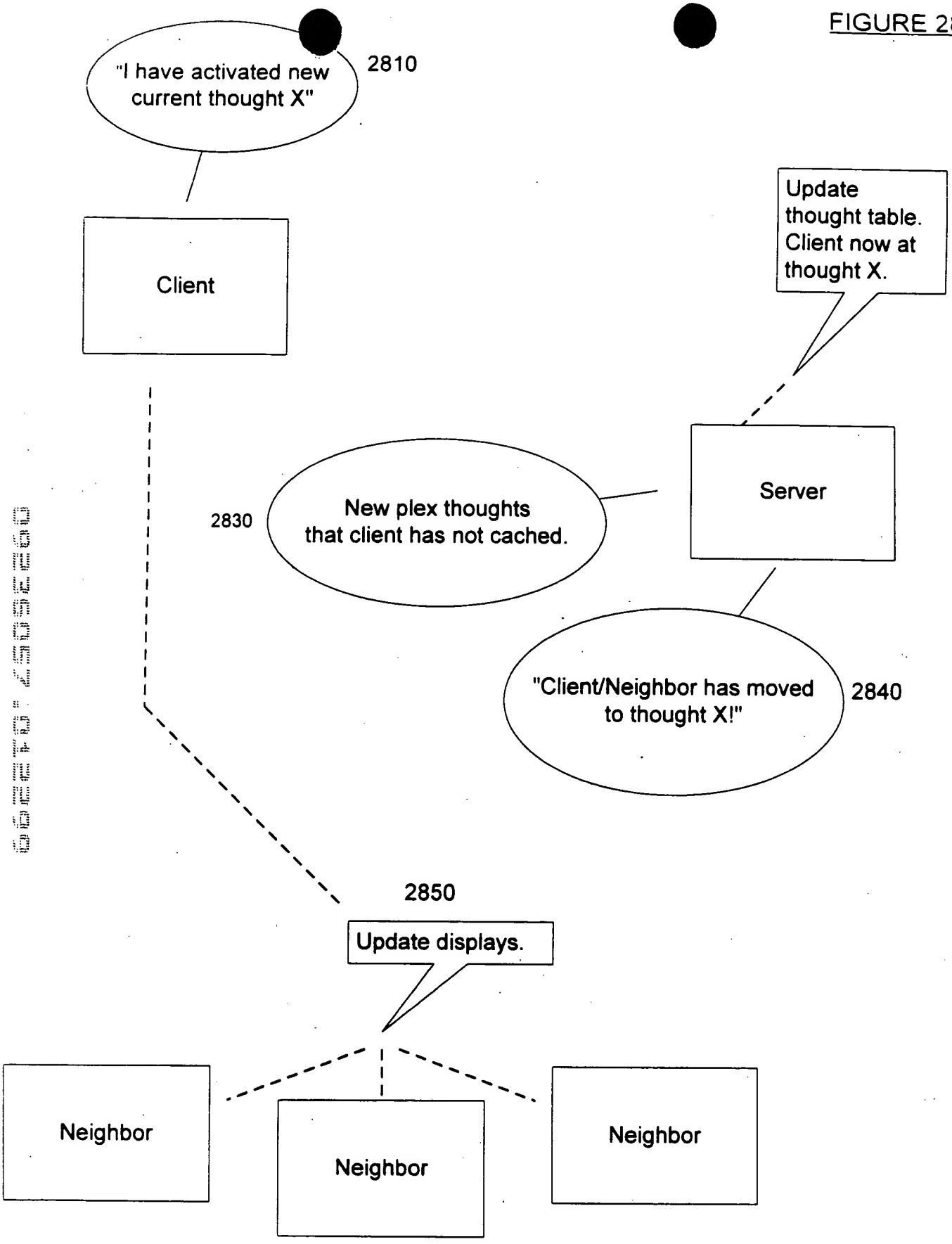
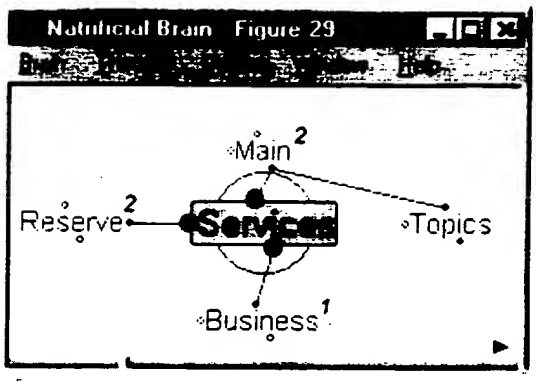
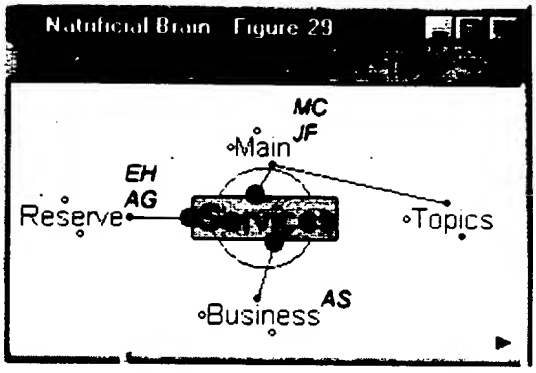


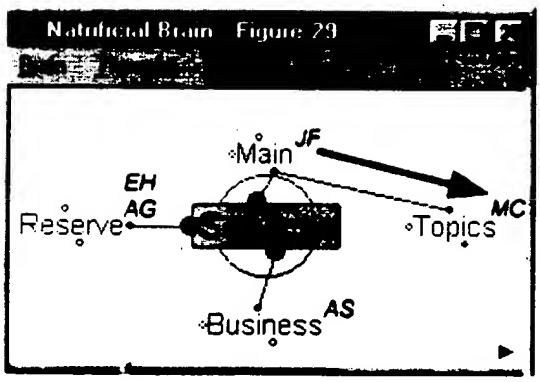
Figure 29



2910

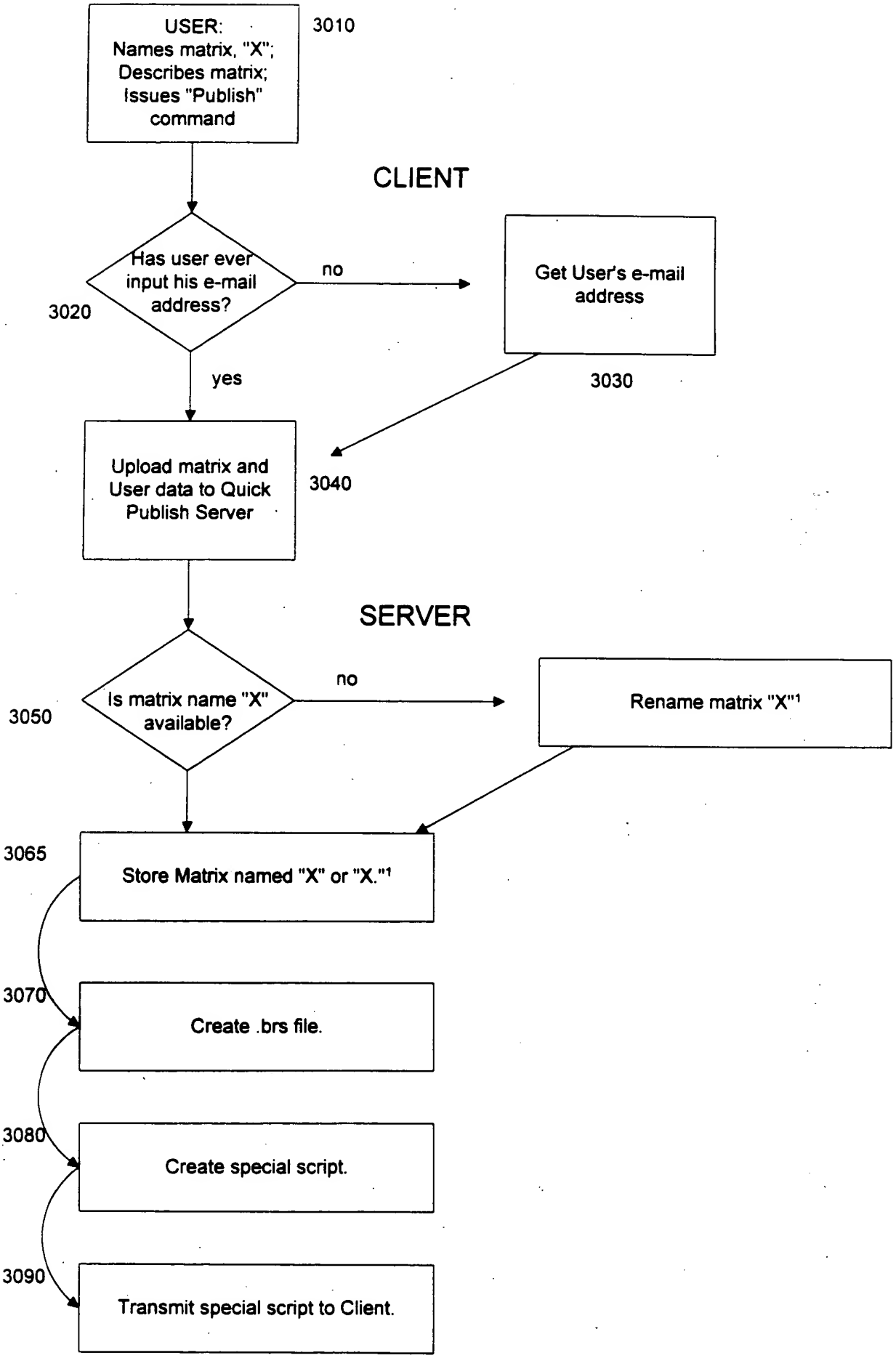
2920





2930

FIGURE 30



ASEGfig30.vsd

Figure 31

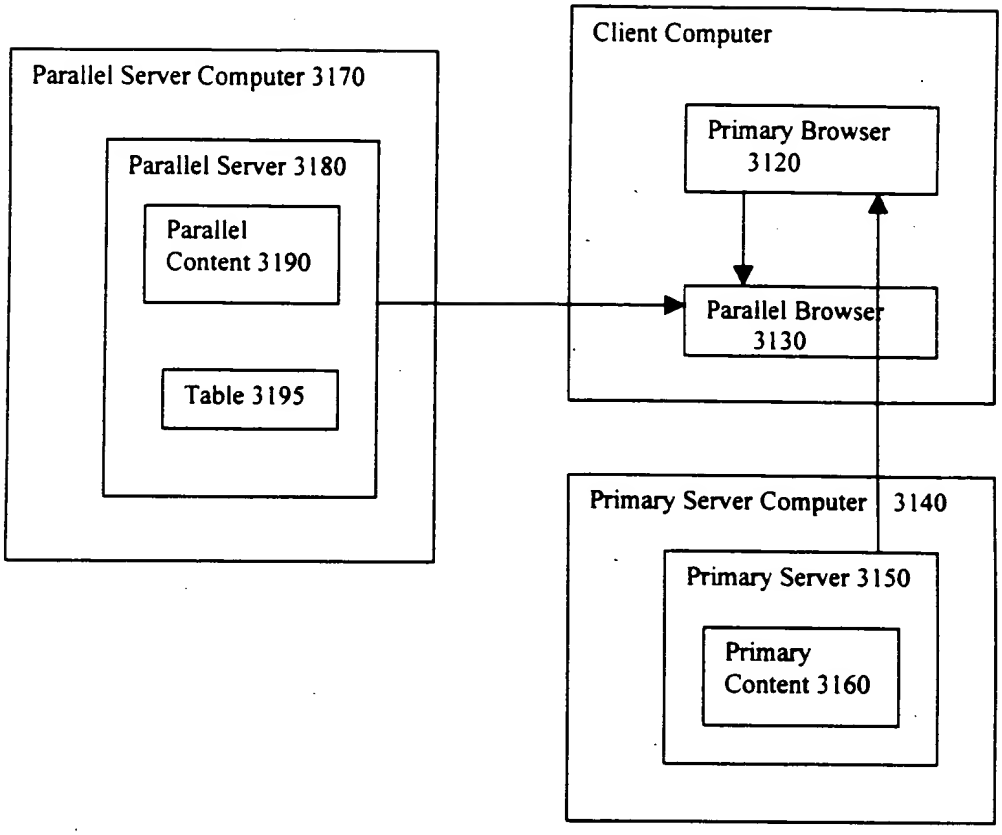
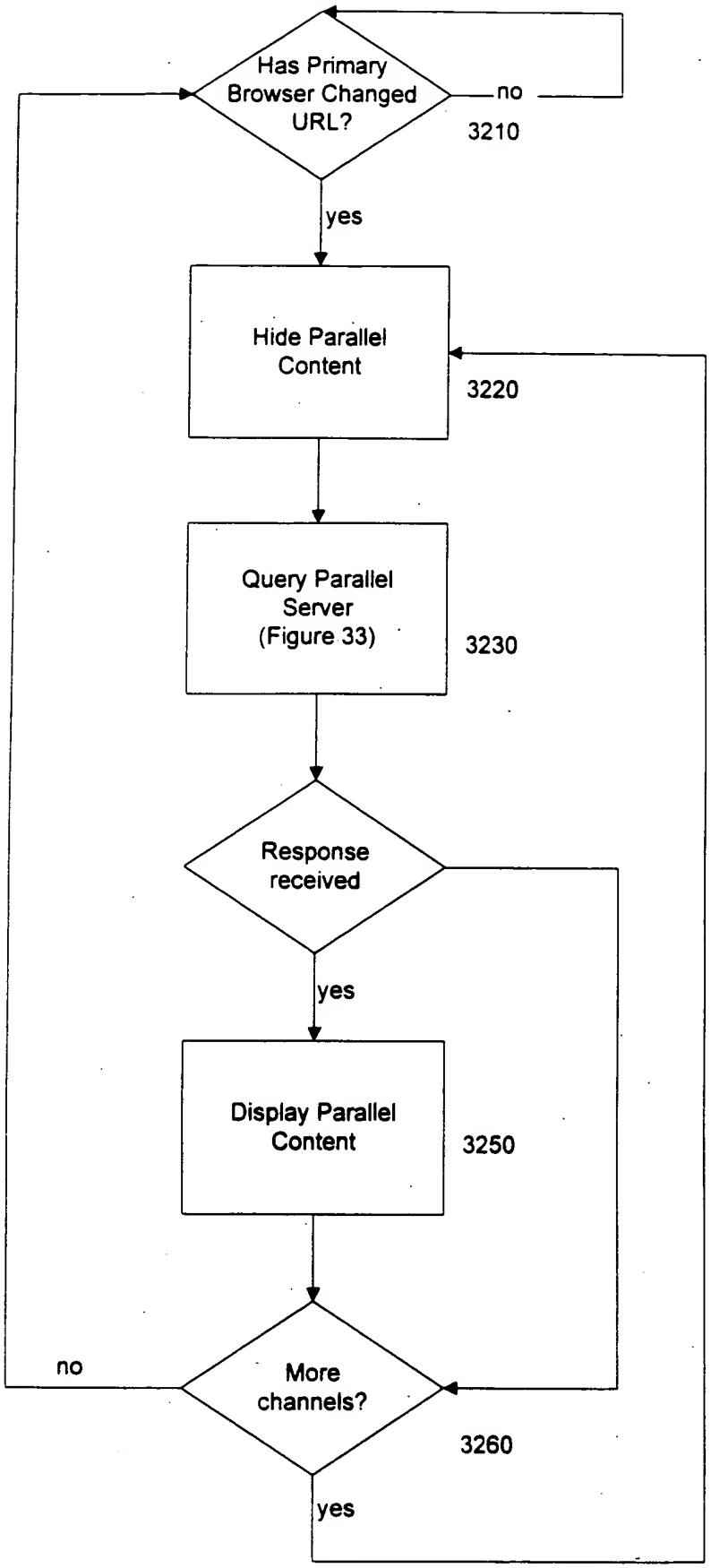
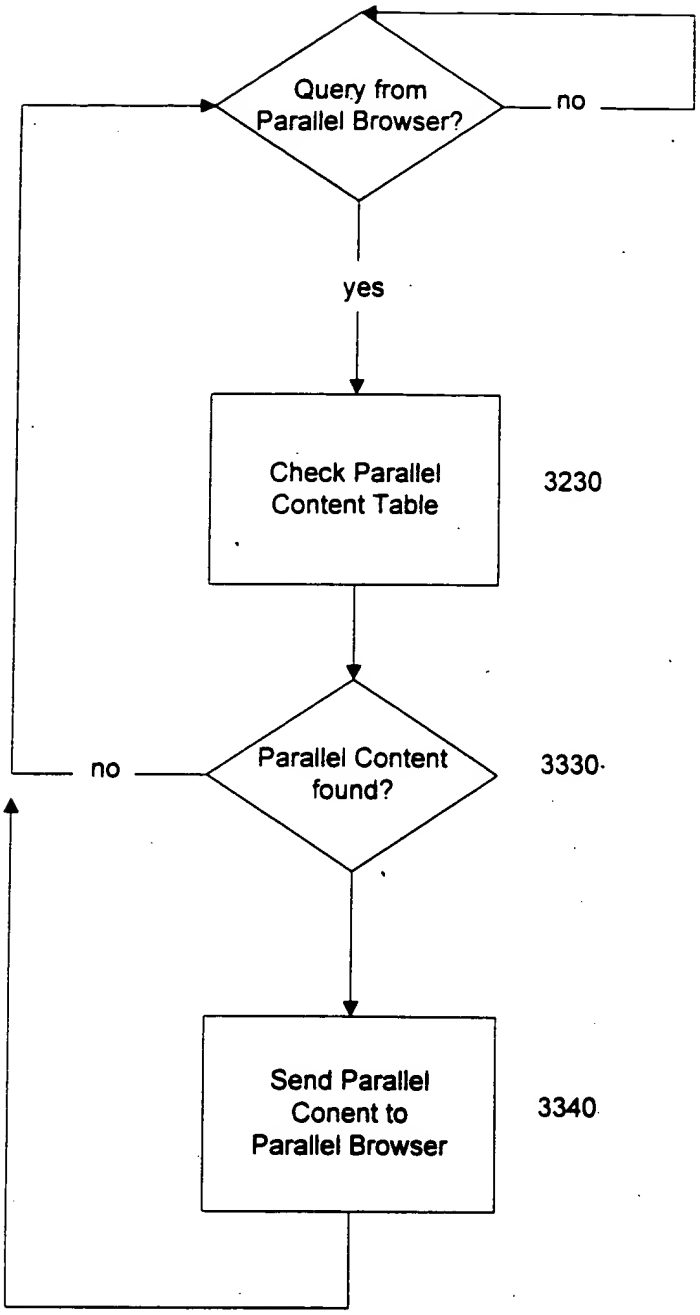

Client Computer
Primary Browser 3120
Parallel Browser 3130
Parallel Server Computer 3170
Parallel Server 3180
Parallel Content 3190
Table 3195
Primary Server Computer 3140
Primary Server 3150
Primary Content 3160

FIGURE 32



ASEGfig32.vsd

FIGURE 33



ASEGfig33.vsd